PatchSim Software Guide

Srini Venkatramanan[*]

August 9, 2018

1 Introduction

Two common modeling frameworks for simulating epidemic dynamics are: ordinary differential equations (ODEs) and agent-based models (ABMs). While the former assumes a homogeneous mixing among the entire population, the latter uses a network to model the heterogeneous mixing among individuals. Though both methods have their own advantages, there is often a need for an approach with an intermediate level of complexity. Metapopulation models bridge the gap between ODEs and ABMs, since they model homogeneously mixing subpopulations connected by a network to model force of infection between subpopulations. See accompanying Model Description document for the underlying mathematical model.

PatchSim is a software code that allows for modeling SEIR dynamics across subpopulations (aka *patches*), connected by a commuting-type flow network (aka *travel matrix*). It has additional features such as vaccinations, stochastic vs deterministic mode, patch-specific parameterization, check-pointing. etc.

2 User Guide

The most recent version of PatchSim is maintained at `https://ndsslgit.vbi.vt.edu/ndssl-software/PatchSim`. To use it, clone the directory, and run `python test_det.py` inside the `test/` folder. It should return no errors, and you should see two new files inside `test/` namely, `test_det.out` and `test_det.log` [1].

There are two basic test configurations (deterministic and stochastic) in the test folder. The deterministic test configuration (`test/cfg_test_det`) looks as follows:

```
PatchFile=test_pop.txt
NetworkFile=test_net.txt
NetworkType=Static

ExposureRate=0.65
InfectionRate=0.67
RecoveryRate=0.4
ScalingFactor=1

SeedFile=test_seed.txt
VaxFile=test_vax.txt
VaxDelay=4
VaxEfficacy=0.5

RandomSeed=12345
StartDate=1
Duration=100
```

[*]Email:vsriniv@bi.vt.edu

[1]If you encounter any errors, contact vsriniv@bi.vt.edu or file a Git issue

```
OutputFile=test_det.out
OutputFormat=Whole
LogFile=test_det.log
```

There are five distinct sections in the configuration file (separated by empty lines for ease of comprehension). We will now take a closer look at each of the keywords and associated input files.

Note: All file paths are relative to the main code (in this case `test/test.py`). All input files below are *space separated* unless otherwise specified. No spaces allowed on either side of the '=' sign in the configuration file.

2.1 Patch & Network Info

- `PatchFile` Contains patch population sizes in the following format for each line:
  `<patch_id> <population_size>`
- `NetworkFile` Contains the travel matrix in the following format for each line:
  `<patch_id_i> <patch_id_j> <time_index> <Theta_i,j>`

  where `<patch_id_i>` and `<patch_id_j>` must be present in the `PatchFile`. `<time_index>` will take different values depending on the `NetworkType` described below. Note that PatchSim expects that Θ as described in the `NetworkFile` obeys row stochasticity, failing which it may lead to errors or incorrect outputs.
- `NetworkType`: PatchSim currently accepts three different NetworkTypes: `Static`, `Monthly`, `Weekly`. Depending on the NetworkType, the `<time_index>` in NetworkFile takes the following values.
  - `Static`: `<time_index>` is always 0.
  - `Monthly`: `<time_index>` takes all values from $[0, 11]$ corresponding to month index (0 = Jan).
  - `Weekly`: `<time_index>` takes all values from $[0, 52]$ corresponding to weeks (0 = Jan 1-7).

All three keywords in this section are mandatory.

2.2 Disease Parameters

- Disease parameters `ExposureRate`, `InfectionRate` and `RecoveryRate` respectively feed into β, α and γ respectively of the dynamics. All the three keywords are mandatory.
- `ScalingFactor` is used to scale the daily infections before printing to the output file. This can be used to represent any of the following: (a) Reported infections, (b) Hospitalizations, (c) Emergency department visits, (d) Deaths. This is an optional parameter, whose default value is set as 1.

2.3 Seeding & Vaccination

- `SeedFile` Contains the (mandatory) seeding schedule in the following format for each line:
  `<day> <patch_id> <seed_cases>`
- `VaxFile` Contains the (optional) vaccination schedule in the following format for each line:
  `<day> <patch_id> <vaccinations>`
- `VaxDelay` and `VaxEfficacy` represent the vaccine delay (in days) and vaccine efficacy (as a probability). Both of these are optional (default vaccine delay is 0, default vaccine efficacy is 1).

2.4 Simulation Time/Style

- `RandomSeed` is an optional argument, which when set automatically triggers stochastic mode of operation. It takes an integer value. (Run `python test/test_stoc.py` to test this feature.)
- `StartDate` takes value from $[0, 365]$ (0 = Jan 1) and denotes the starting date (for the travel matrix)
- `Duration` is the simulation duration in days (positive integer)

Both keywords are mandatory (although `StartDate` is not used if `NetworkType` is `Static`).

2.5 Outputs

- `OutputFile` is produced at the end of simulation. It contains the epicurves of each patch in the following format for each line:
  `<patch_id> <cases_0> <cases_1> ... <cases_T>` where T is the duration of the epidemic.

- `OutputFormat` (optional) specifies if PatchSim must produce integer (`Whole`) or floating point (`Fractional`) values in output time series. Default value is `Whole`, which will produce integer outputs (floor).

- `LogFile` (optional) file for basic logging messages with timestamp.

3 Advanced Features

In addition the above listed attributes, PatchSim supports two additional advanced features described below:

- **Checkpoints**: PatchSim allows you to load and save the state array (values of S,E,I,R and V states for all patches). User can run a simulation till day T, save state, and restart new simulation by loading the saved state. This is accomplished by the following entries in the config file:

  ```
  LoadState=True
  LoadFile=checkpoint1.npy
  SaveState=True
  SaveFile=checkpoint2.npy
  ```

  The above code, loads state from `checkpoint1.npy`, runs the simulation, and then saves final state to `checkpoint2.npy`. When not being used, `LoadState` and `SaveState` must be set to `False`. (Run `python test/test_stopstart.py` to test this feature.)

- **Patch-specific parameters**: The disease parameters `ExposureRate`, `InfectionRate` and `RecoveryRate` are usually listed in the Config file. If the user intends to use heterogeneous parameters for the patches, then one can replace the above three lines with a `ParamFile` attribute, pointing to a file.

  The file is space separated and contains a header line `id beta alpha gamma` followed by patch specific parameters for each patch. Note that all patch ids present in the patch populations file must be listed. Also, `beta, alpha, gamma` correspond to `ExposureRate, InfectionRate, RecoveryRate` respectively. (Run `python test/test_paramfile.py` to test this feature.)

4 Limitations

PatchSim is under constant development. The following are some of its current limitations, which we are hoping to address in subsequent releases.

- **Flow matrix**: The current flow matrix interpretation is best suited for commuting type datasets, and assumes the individual spends **entire day** at the work patch. If you want to restrict to working hours, the travel matrix needs to be appropriately scaled.

  For other types of flows (air travel, non-work activity trips), one needs to explicitly process the travel matrix accounting for stay duration. Further, population migration cannot be handled by this model.

- **Disease Model**: Currently PatchSim supports only SEIR disease model with vaccination. We hope to incorporate other disease models, including vector-borne models soon.

Feedbacks are welcome. Any feature requests or bug reports can be filed filed as Git issues at: `https://ndsslgit.vbi.vt.edu/ndssl-software/PatchSim`.

5 Acknowledgments

PatchSim development has been guided by inputs from the following folks at NDSSL (in alphabetical order): Jiangzhuo Chen, Arindam Fadikar, Sandeep Gupta, Bryan Lewis, Madhav Marathe, Anil Vullikanti.

Version History

- **23 Nov, 2017**: First version released.
- **27 Nov, 2017**: Ensured python3 compatibility.
- **21 Feb, 2018**: Checkpoints feature added.
- **23 Feb, 2018**: Patch-specific parameters allowed.
- **08 Aug, 2018**: Stochastic mode (through RandomSeed) added.